Resources for Modifying LAMMPS

Steve Plimpton
Sandia National Labs
sjplimp@sandia.gov

4th LAMMPS Workshop
August 2015 - Albuquerque, NM

Sandia National Laboratories is a multi-program laboratory managed and operated by Sandia Corporation, a wholly owned subsidiary of Lockheed Martin Corporation, for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-AC04-94AL85000.
Presentation: SAND2015-6465C

Before you start writing code ...

Q: Should LAMMPS do it?

- Better as pre- or post-processing?
- Web site: pre/post tools, offsite tools, Pizza.py, etc

Before you start writing code ...

Q: Should LAMMPS do it?

- Better as pre- or post-processing?
- Web site: pre/post tools, offsite tools, Pizza.py, etc

Q: Can LAMMPS already do it?

- Check the documentation and web site
  - http://lammps.sandia.gov
  - http://lammps.sandia.gov/glossary.html
  - http://lammps.sandia.gov/doc/Manual.html
  - http://lammps.sandia.gov/doc/Section_commands.html
  - http://lammps.sandia.gov/doc/Section_howto.html
  - New-format doc pages have search box
  - And an index: http://lammps.sandia.gov/doc/genindex.html
  - Web pages: other tools, papers by others, etc

Before you start writing code ...

Q: Has someone else wanted to do this?

- Search the mail list = 55K messages
  - http://lammps.sandia.gov/mail.html
  - http://lammps.sandia.gov/threads/threads.html
  - Google: lammps-users thermostat Lowe
    1st hit: lammps.sandia.gov/threads/msg20748.html
    3rd hit: SourceForge.net: LAMMPS: lammps-users
    5th hit: Thermostats at Lowe's (www.lowes.com)

Q: What does the LAMMPS community think?

- Post a "how can I do this" message to the mail list
  - email to lammps-users@lists.sourceforge.net

6 ways to modify LAMMPS: easy to hard

1. Input script options
2. Write Python code
3. Couple to another code
4. Small changes to existing customizable files
5. Write code for a new style
6. Create a new package

6 ways to modify LAMMPS: easy to hard

1. Input script options
2. Write Python code
3. Couple to another code
4. Small changes to existing customizable files
5. Write code for a new style
6. Create a new package

- Details on all of these in past workshop and tutorial slides
  - http://lammps.sandia.gov/workshops.html at bottom
  - http://lammps.sandia.gov/tutorials.html
  - see my talks
- Developers manual (brief!)
  - doc/Developer.pdf
  - diagram of class hierarchy
  - pseudo-code & explanation of how a timestep works

#1 - Input script options

- Input script syntax is a simple programming langauge
- Programming-like commands:
  - if (then else), jump, next, label, include, print
  - shell command to invoke other programs
- Variables
  - can store one or more strings/numbers
  - can store formulas applied to scalars or per-atom values
    - formulas can access output of many other commands
    - invoked immediately in input script
    - invoked periodically during a simulation
    - can be embedded in immediate variables
  - can read values from files
  - can be used as input to various commands
    - fixes, computes, averaging fixes, dump/thermo output
  - understand Section howto 15 on flavors of command outputs

#2 - Using Python with LAMMPS

- Manual Chapter 11: Python Interface to LAMMPS
- Write a Python script that invokes LAMMPS
  - instantiate one or more LAMMPS instances
  - invoke LAMMPS input script commands
  - invoke functions in LAMMPS library interface (extensible)
  - grab LAMMPS data, alter it, pass it back
  - see python dir of distro for examples

#2 - Using Python with LAMMPS

- Manual Chapter 11: Python Interface to LAMMPS
- Write a Python script that invokes LAMMPS
  - instantiate one or more LAMMPS instances
  - invoke LAMMPS input script commands
  - invoke functions in LAMMPS library interface (extensible)
  - grab LAMMPS data, alter it, pass it back
  - see python dir of distro for examples
- New! Call Python code from your input script
  - new python command
  - define a Python function in input script or file
  - associate with python-style variable
  - invoke Python function
    - immediately in input script
    - whenever variable is evaluated
  - can pass LAMMPS data to Python function
  - Python function can callback to LAMMPS thru lib interface
  - use case examples in Section 11.2

#3 - Couple to another code

- Section howto 6.10: Coupling LAMMPS to other codes
  - wrap the other code in a compute or fix
  - example: Voro++ library called by compute voronoi/atom
  - pass atom coordinates
  - Voro++ returns Voronoi tesselation
  - compute voronoi/atom outputs the per-atom results
  - when build LAMMPS, link with Voro++
- Section howto 6.19: Library interface to LAMMPS
  - C-style, so can be called from C++/C/Fortran/Python
  - easy to extend, just add functions to library.cpp/h
  - add wrapper method to python/lammps.py for Python
  - enables another code to invoke LAMMPS
  - enables Python wrapper to invoke LAMMPS and other code, pass info between

#3 - Couple to another code (continued)

Additional tools that may be useful:

- examples/COUPLE
  - lib for parallel data communication between 2 codes
  - example: used by LIGGGHTS
- Fix external command, see doc/fix_external.html
  - trigger LAMMPS to callback to driving program every N steps
  - example: quantum code drives LAMMPS, provides DFT forces

#4 - Make small changes to existing files

Look for customize comments in appropriate src file

#4 - Make small changes to existing files

Look for customize comments in appropriate src file

1. Adding keywords to thermo_style output
   - see thermo.cpp
   - complicated calculation better done as new Compute
2. Adding keywords for per-atom data vectors/arrays
   - see compute_property_atom.cpp
   - allows its use in all other commands
     - dump, fix ave/spatial, atom-style variables, etc
3. Adding new functions to equal-style and atom-style variables
   - see variable.cpp
   - math functions, special functions, math operators, etc
   - follow syntax rules for args of similar functions

#5 - Write code for a new style

- A style is a class derived from a parent style
- 95% of LAMMPS code base is add-on styles
- Manual Chapter 10: Modifying & Extending LAMMPS

#5 - Write code for a new style

- A style is a class derived from a parent style
- 95% of LAMMPS code base is add-on styles
- Manual Chapter 10: Modifying & Extending LAMMPS
- 16 kinds of styles (ls src/style.*h)
  - particle types = atom style
  - force fields = pair style, bond, angle, dihedral, improper
  - long range = kspace style
  - fix = fix style = BC, constraint, time integration, ...
  - diagnostics = compute style
  - geometric region = region style
  - input/output = dump style, reader style
  - minimizer = min style
  - integrator = integrate style
  - input command = command style = read_data, velocity, run

#5 - Write code for a new style

- A style is a class derived from a parent style
- 95% of LAMMPS code base is add-on styles
- Manual Chapter 10: Modifying & Extending LAMMPS
- 16 kinds of styles (ls src/style.*h)
  - particle types = atom style
  - force fields = pair style, bond, angle, dihedral, improper
  - long range = kspace style
  - fix = fix style = BC, constraint, time integration, ...
  - diagnostics = compute style
  - geometric region = region style
  - input/output = dump style, reader style
  - minimizer = min style
  - integrator = integrate style
  - input command = command style = read_data, velocity, run
- Find an existing style file similar to what you want to do
- Create fix_foo.cpp/h, drop in src dir, re-build, that's it

#6 - Create a new package

- A package is simply a collection of related style files
- LAMMPS developers maintain STANDARD packages
  - written in LAMMPS-style syntax, e.g. error messages
  - src/MANYBODY, src/KSPACE, src/VORONOI, etc
- Contributors maintain USER packages
  - written however you like (within reason)
  - USER-MISC contains single-file user contributions
  - src/USER-OMP, src/USER-EFF, src/USER-INTEL, etc
- Packages can also have build and external dependencies
  - src/PACKAGE/Install.sh tailors build process
  - lib/package dirs (colvars, gpu, kim, voronoi, etc)
  - auxiliary libs are pre-built separately from LAMMPS

Contributing your new code to the LAMMPS distro

- Why release it as part of main LAMMPS?
  - open source philosophy
  - fame and fortune, name on author page and in source code
  - acquire users of your feature
    - find and fix bugs
    - extend its functionality
    - become collaborators
- Read Section modify 15 first (even before writing the code!)
  - title: Submitting new features for inclusion in LAMMPS

Contributing your new code to the LAMMPS distro

- Why release it as part of main LAMMPS?
  - open source philosophy
  - fame and fortune, name on author page and in source code
  - acquire users of your feature
    - find and fix bugs
    - extend its functionality
    - become collaborators
- Read Section modify 15 first (even before writing the code!)
  - title: Submitting new features for inclusion in LAMMPS
- Key points for a speedy release:
  - Doc pages for new commands, in LAMMPS format (doc/*.txt)
  - Avoid changes (if at all possible) to core LAMMPS files
    - ask ahead if want to avoid being asked to re-write code
- Then email us the files
  - Tarball to Steve with src,doc,examples,etc dirs
  - Git request to Axel

That's all

Questions?